Projektopgave forår 2015 - del 2
02325 - Datakommunikation

Projektnavn:

Gruppe nr: 06

Afleveringsfrist: Tirsdag d. 14 kl 18:00

Den rapport er afleveret via Campusnet(der skrives ikke under).

Denne rapport indeholder *xx* sider inklusiv denne side.

S144847, Nielsen, Jon Tvermose

Kontaktperson(Projektleder)

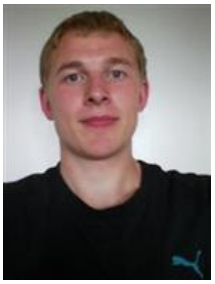
S125015, Berthold, Ebbe Bjerregaard

S144855, Olsen, Camilla Braae

S144875, Jepsen, Jacob Worck

S144843, Mikkelsen, Christian Flygare Carlend

S144826, Frantsen, Tobias



Indholdsfortegnelse

Forord

Denne rapport består af to projekter - det ene projekt er program som forbinder til en ftp-server via en FTP-klient, for at overføre filer fra serveren til klienten. Dette sker gennem en protokol som er skabt fra bunden.

Det andet projekt er et program som styrer en vægt (evt. vægtsimulator) i forhold til en foruddefineret afvejningsprocedure. WCU har via tekstfiler adgang til en række informationer om lagerbeholdning, varenavne, operatørnavne m.m. WCU'en sørger også for at logge informationer om en afvejning når sådan en er foretaget.

Gennemgangen af begge projekter i rapporten er for sin vis den samme, dog er der forskel mellem projekterne i design-afsnittet, da der i FTP-projektet er lagt vægt på at forklare hvad der sker mellem klienten og serveren, når der bliver sende en kommando eller fil imellem dem.

I vægtkontrol-projektet er der lagt vægt på kommandoerne der bliver sendt til vægten er korrekte, og at afvejningsproceduren tager højde for så mange fejlsituationer som muligt.

Konfiguration

FTP programmet skal modtage 6 start argumenter når programmet startes. Dette kan gøres via kommandoprompt hvis .jar filen afvikles eller via Eclipse's funktion.

Argumenterne er følgende:

1. FTP host
2. FTP port (21)
3. Username
4. Password
5. Zybo host
6. Zybo port

WCU programmet skal modtage to argumenter når programmet startes. Argumenterne er følgende:

1. Host
2. Port

FTP Klient

Analyse

Kravspecifikation

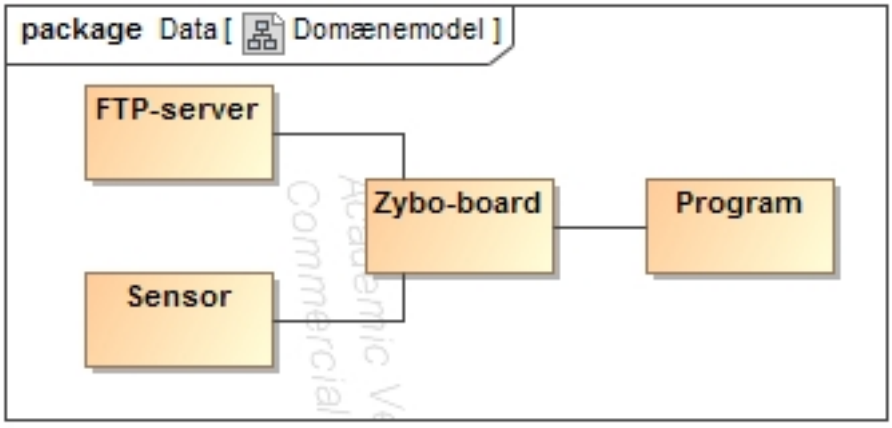
Funktionelle krav

- FTP-klient skal kunne hente en eller flere filer fra ftp-server
- Sende kommandoer til sensorere på et zybo-board
- Programmet skal have en menu
- FTP-klient skal kunne forbinde til en ftp-server på et zybo-board
- Både LIST og RETR skal kunne sende sendes til FTP-serveren
- Svaret på kommandoerne skal vises til brugeren
- Programmet skal laves i samme eclipseprojekt som WCU'en
- Skal kunne sende kommandoer til et zybo-board

Non-funktionelle krav

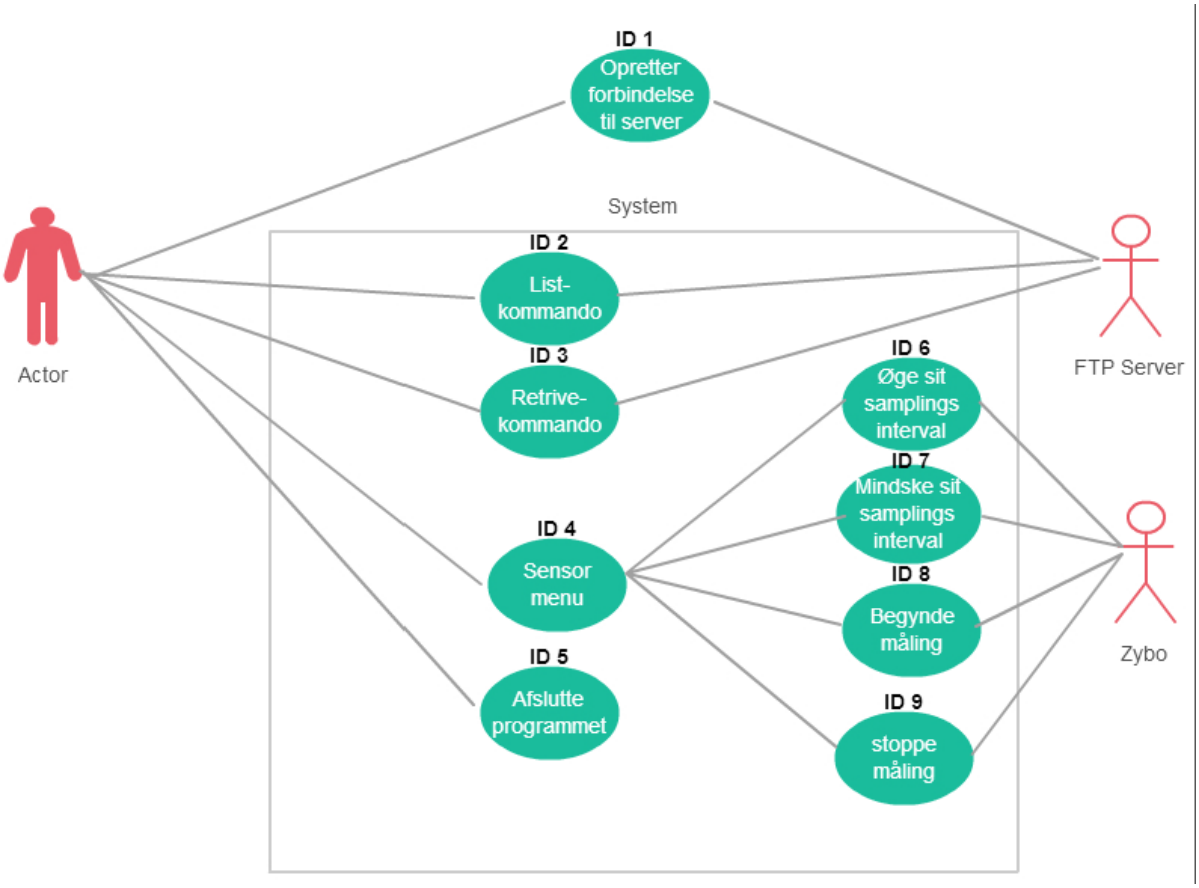
- FTP-klienten skal være simplificeret

Domæne Model



I ovenstående figur ses domænemodellen. Som det ses på figuren, forbinder programmet til zyboboardet for både at sende kommandoer til FTP-serveren og sensorerne. Dog er disse lavet på to vidt forskellige måder, da FTP-serverens kommandoer bliver sendt via en FTP-klient som formaterer kommandoerne til brug i FTP-serveren. Sensorerne på zybo-boardet kræver ikke nogen formatering, så derfor er der brugt en almindelig TCP forbindelse til at sende kommandoer til disse. Dette ses bedre i figur use-case diagrammet.

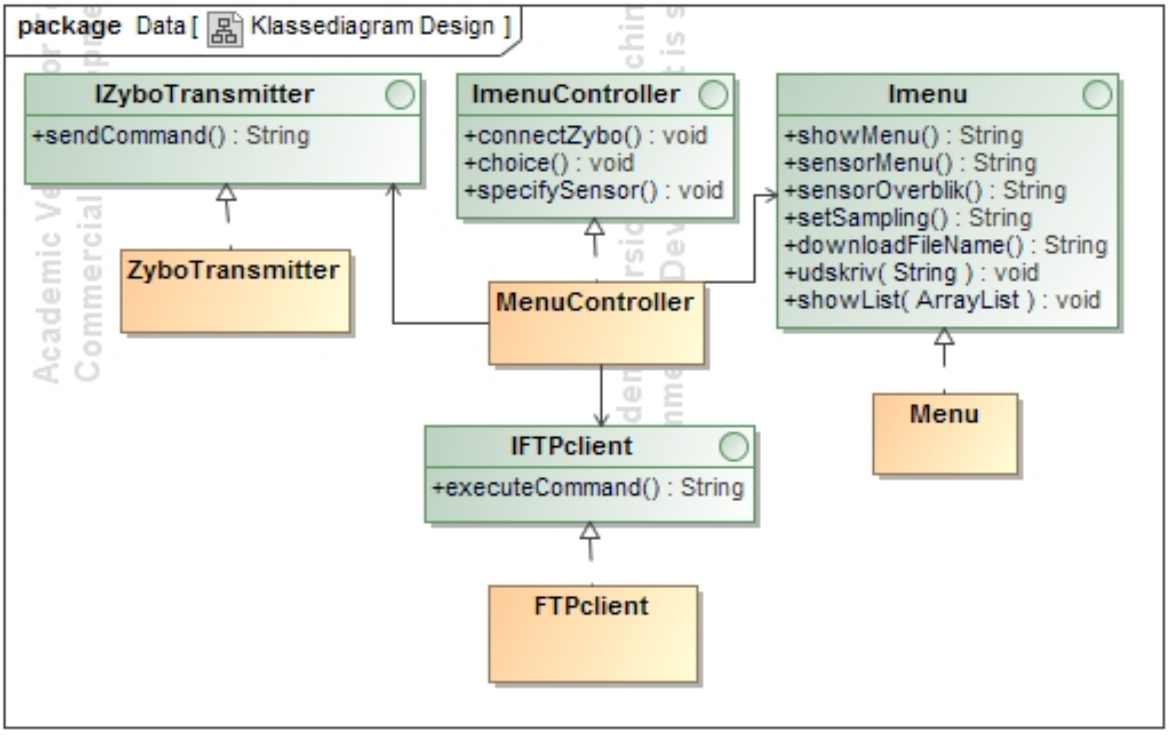
Use Case Model



Ovenstående figur viser de forskellige use-cases der er blevet anvendt til opbygningen af FTP klienten. Modellen er levet på den måde, at det der er uden for boksen system, er noget der kun sker en gang, når programmet startes. Alt inde i boksen system kører hele tiden, og man kan blive ved med at anvende disse menuer. Under bilag ligger mere detaljerede beskrivelse af use case modellen.

Design

Klassediagram



Her ses klassediagrammet over FTP-delen. I diagrammet ses det at MenuController styrer programmets logik ved at forbinde til alle interfaces i programmet. Hver klasse har implementeret et interface, som indeholder alle metoder der bruges for at benytte klasserne.

FTP-klient

FTP står for "File Transfer Protocol" og er en klient – server protokol der benyttes til at overføre filer.

I dette projekt er der bl.a. stillet til opgave at implementere en FTP-klient via Java's eget bibliotek, som skal kunne forbinde til en FTP-server og udføre LIST og RETR kommandoen. LIST kommandoen får FTP-serveren til at udprinte information om det nuværende directory, specifikt hvilke filer dette indeholder. RETR kommandoen kopierer en given fil fra FTP-serveren til harddisken. RETR er implementeret således at filnavnet på filen der ønskes downloadet/kopieret til harddisken, skrives ind i consollen fra brugerens side.

Forbindelsen til FTP-serveren muliggøres ved implementeringen af en simpel FTP-klient. Ingen yderligere software er downloadet.
LIST- og RETR-funktionen er implementeret i executeCommand()-metoden. WireShark har gjort det muligt at følge FTPs procedure, når FTP-klienten laver forespørgsler til FTP-serveren, og dette har været til stor gavn da executeCommand() blev implementeret. Begge metoder udfører nemlig samme kommandoer til FTP-serveren inden dataforbindelsen, så dette kunne fusioneres sammen i én metode.

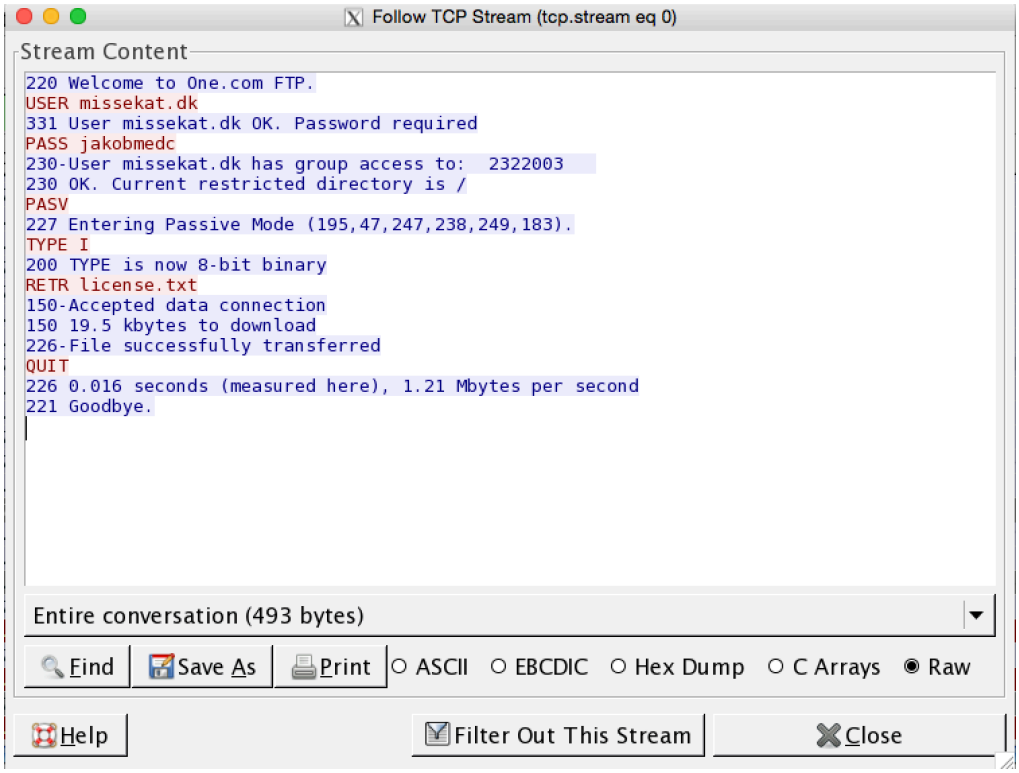
```
public String executeCommand(String host, int port, String command,
```

Ovenstående billede viser et udsnit fra executeCommand(..)'s parameter. String command afgør hvorvidt det er LIST eller RETR kommando og længere nede i metodens kode, switches der på om String command = "LIST" eller String command = "RETR". Værdien af String command afgøres i MenuControlleren.

```
switch (command.toUpperCase()){
case "LIST":
    sendLine("LIST", bw);
    getDataLIST(p[0], Integer.parseInt
    reply = "Listen er hentet.";
    break;
case "RETR":
    sendLine("TYPE I", bw);
    readLine(br);
    sendLine("RETR " + fileName, bw);
```

På ovenstående billede vises den del af koden, der gør executeCommand() unik. Hele opsætning til FTP-serveren er ens for både LIST og RETR, men når der switches på String command adskilles de.
Hvis det senere hen ønskes at implementere en upload-funktion fra FTP-klientens side kan executeCommand() genbruges, grundet ligheden af de første FTP-kommandoer til FTP-serveren. Det kræver blot at switchen modtager endnu et argument, at switche på.

FTP WireShark syntaks:



Ovenstående billede er et udsnit fra WireShark hvor der blev downloadet en fil (license.txt) fra FTP-serveren (ftp.missekat.dk). FTP-serverer svarer tilbage med tal, som kan indikere alt lige fra forbindelse, mistet forbindelse, fil-overføres osv. På billedet af samtalen mellem FTP-klienten og FTP-serveren er det FTP-serveren der har blå skrift. Her ses det tydeligt, at der ved oprettelse af forbindelse til host' printes kode 220, som signalerer at serveren er klar til at modtage et evt. brugernavn og adgangskode.

I eksemplet sendes fra klientsiden "*USER missekat.dk*". Såfremt brugernavnet godkendes sendes en meddelelse til klienten, at en kode er påkrævet før videre adgang tillades. Her sendes der "*PASS jakobmedc".* Syntaksen for at sende en kommando til FTP servere er defineret i RFC959.

Længere nede i processen sendes en "*PASV*" kommando til FTP-serveren. "*PASV*" kommandoen beder serveren om at gå i passive mode. Det betyder at det er klienten der skal oprette en forbindelse til FTP-serveren for at få overført data. Alternativet er en "PORT" kommando hvor serveren vil oprette forbindelse til klienten, og derefter sende data over den forbindelse. PASV-kommandoen er valgt idet den er mere sikker at benytte for klienten.

Serveren vil svare en PASV-kommando ved at sende et svar til klienten, indeholdende IP-adressen til serveren, samt to tal der muliggør at beregne af det nye port nummer. I ovenstående eksempel er der tallene: (195,47,247,238,249,183). De første fire tal er ip-adressen 195.47.247.238. De sidste to tal, 249 og 183, anvendes til at beregne data-porten. Det gøres på følgende måde: 249*256+183 = 63.927. I dette tilfælde anvendes port 63927 til dataforbindelsen. Der oprettes herefter en ny TCP-forbindelse til serveren med den pågældende ip-adresse og den nye port. Hvis forbindelsen bliver accepteret af serveren, svarer denne med en besked "*150-Accepted data connection*". Bemærk at dette svar bliver sendt på den oprindelige forbindelse (Kommando porten), og altså ikke på den nye forbindelse der er blevet oprettet. Først efter denne besked er modtaget, påbegyndes overførslen af data på dataforbindelsen. Når overførslen er slut sendes endnu en besked "*226-File succesfully transfered*" på den oprindelige forbindelse.

I systemet er det valgt at lukke alle forbindelser til FTP-serveren efter hver kommando er udført. Dette er valgt idet der fra serverens side kan være opsat en begrænsning på hvor længe en forbindelse kan være inaktiv før den lukkes. Forbindelsen til serveren lukkes fra klientensiden ved at sende kommandoen "*QUIT*".

Inspiration til implementeringen af FTP-klienten, indeholdende LIST og RETR kommando, kan findes i Bilag.

Sensor

I opgaven stillet, er der krav om at programmet kan kontakte nogle sensorer på et zybo-board. Dette krav er løst via en TCP socket til boardet, samt en intuitiv menu i programmet.

```
**************************************
Indtast 1 for list-kommando
Indtast 2 for retrieve-kommando
Indtast 3 for sensor-menu
Indtast 4 for at afslutte programmet
**************************************

3
```

Når programmet startes vil der først blive tjekket for om der er forbindelse til zybo-boardet via den socket*indsæt linjetal* der er oprettet i MenuController. Dette tjek sker i *indsæt linje talt* i ZyboTransmitter.

Derefter får brugeren et valg i hovedmenuen, hvor der kan vælges om der skal sendes kommandoer til FTP-klienten eller til om sensorerne på boardet skal kontaktes. I denne menu kan programmet også afsluttes.

```
**************************************
Sensor overblik for alle sensorere i systemet
Indtast tal for hvilken sensor du vil bruge
Tast 1 for sensor 1
Tast 4 for sensor 4
Tast 222 for sensor 222
Tast 7 for sensor 7
Tast e for at gå tilbage til hovedmenu
**************************************

1
```

I den næste menu kan brugeren vælge hvilken sensor der skal sendes kommandoer med. Dette er vigtigt, da programmet skal vide hvilken sensor den snakker med på boardet, for at kunne sende kommandoen korrekt.

```
**************************************
Indtast hvilken funktion der skal udføres
I: Bed sensor om at ændre sit samplingsinterval.
S: Bed sensor om at begynde måling.
B: Bed sensor om at stoppe måling.
e: Gå tilbage til hovedmenu.
**************************************

I
**************************************
Hvor meget vil du ændre samplingsintervallet? (eks: 12300)
**************************************

10000
```

Når dette vælges, vil en ny menu bede om hvilken kommando der skal vælges. Valget om at starte eller stoppe målinger i sensoren vil aktivere kommandoen til sensor, hvor et valg om at ændre samplingsintervallet vil aktivere endnu en menu, som spørger om hvad intervallet skal sættes til.

```
91      public void specificSensor(String sensor) throws NumberFormatException, IOException {
92          String input = menu.sensorMenu();
93          if(input.equalsIgnoreCase("I")) {
94              String sampling = menu.setSampling();
95              menu.udskriv(zbtr.sendCommand(input, Integer.parseInt(sensor), sampling));
96          } else {
97              menu.udskriv(zbtr.sendCommand(input, Integer.parseInt(sensor), null));
98          }
```

Da dette er to forskellige valg, vil ZyboTransmitter have to forskellige parametre, da start/stop-kommandoen ikke skal sende et samplingsinterval med til sensoren.

Der er designet en protokol til at kommunikere med Zyboardet. Protokollen har følgende syntaks: ”Kommando Sensornummer Parameter”. Kommando kan have følgende værdier:

- *S – Start måling*
- *B – Stop måling*
- *T – Returner måling*
- *I – Sæt samplingsinterval*

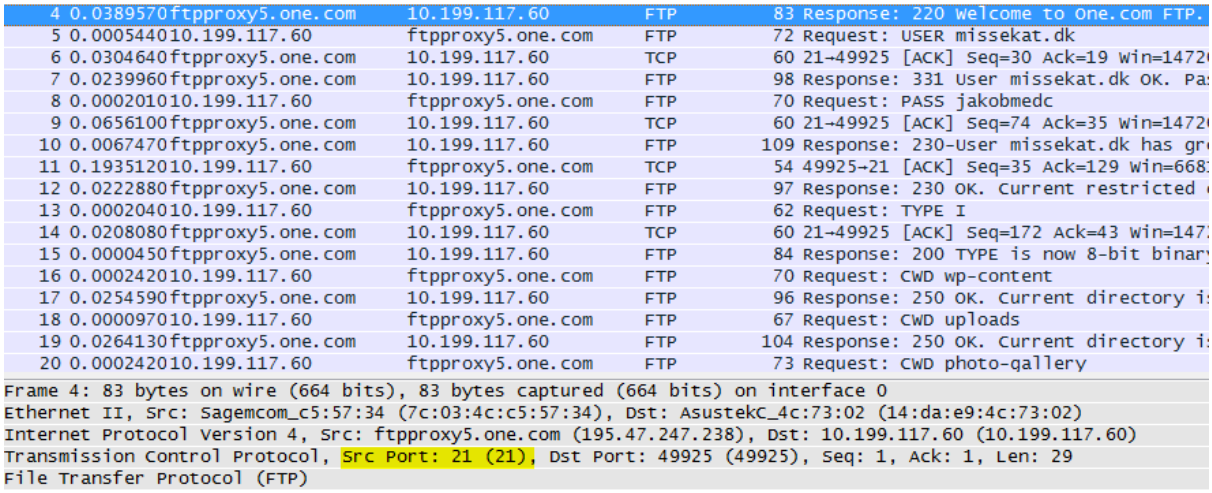
Sensornummer referer til hvilken sensor kommandoen skal udføres på, og består af et tal. I programmet er der antaget Zyboboardet har 4 sensorere, nemlig 1, 4, 7 og 222.

Parameter der medsendes benyttes kun når der skal ændres samplingsinterval. Hvis eksempelvis der ønskes at ændre samplingsinterval til 12000 på sensor 222, skal kommandoen ”*I 222 12000*” sendes. Der vil herefter blive returneret et svar fra Zyboboardet om hvorvidt kommandoen er udført eller fejlet.

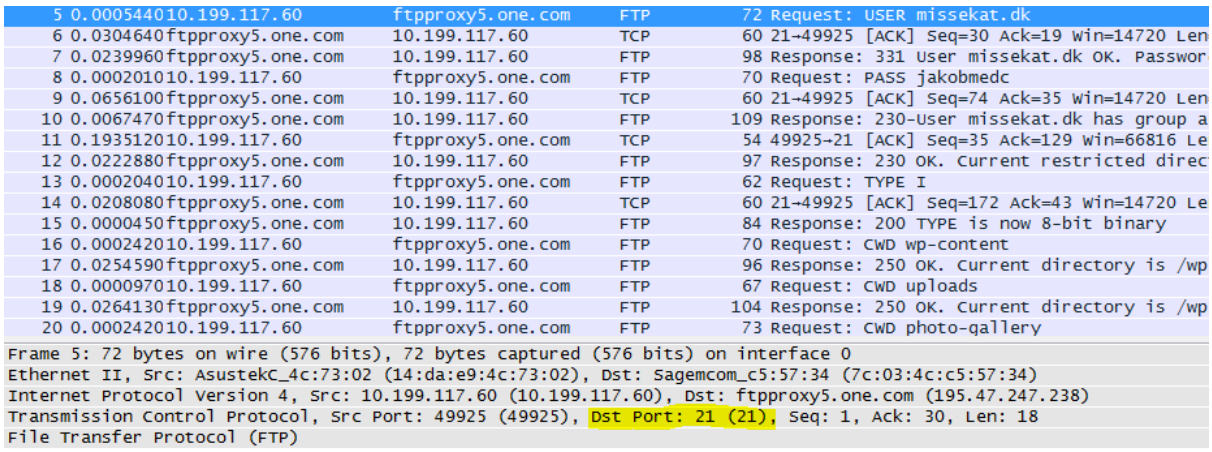
Der er ydermere lavet en metode (*ZyboTransmitter.getSensors()*) der henter en liste af sensorer fra Zyboboardet. Denne metode er benyttes dog ikke i programmet, idet der ikke har været et fungerende program på Zyboboardet at teste implementationen op imod. Her er det antaget at Zyboboardet kan modtage kommandoen ”*LIST*” og derefter returnerer en linje for hver sensor der findes på Zyboboardet. Når alle sensorer er kommunikeret sender Zyboboardet en sidste linje med teksten ”*END*”.

Test

FTP er en protokol der anvender port 21 og 20, til henholdsvis at sende kommandoer og data mellem klienten og serveren. Den sender kommandoer over port 21 og data over port 20, men i nogle tilfælde foregår al kommunikation igennem port 21. Dette testes klienten for, ved hjælp af wireshark.



På billedet oppe ovenfor, kan det ses at FTP serveren svarer tilbage på port 21. Herefter svarer klienten også tilbage på samme port, hvor der bliver sendt et brugernavn til serveren. Dette kan ses på billedet nedenunder:



Ud fra disse to billeder, kan de ses at der er en forbindelse mellem klienten og serveren på port 21. Denne kommando bruges udelukkende til kommandoer.

***Positiv test** - vis Directory og hent fil, som gemmes på harddisken:*

Programmet startes op og LIST funktionen vælges igennem metodekaldet executeCommand(…);

```
**************************************
Indtast 1 for list-kommando
Indtast 2 for retrieve-kommando
Indtast 3 for sensor-menu
Indtast 4 for at afslutte programmet
**************************************

1
**************Start***************
drwx--x--x    6 2320585    2322003          2048 Apr 13 10:32 .
drwx--x--x    6 2320585    2322003          2048 Apr 13 10:32 ..
-rw-r--r--    1 2320585    2322003           236 Apr 30  2014 .htaccess
-rw-r--r--    1 2320585    2322003             0 Aug 11  2014 Ny fil.html
drwxr-xr-x    2 2320585    2322003          2048 Aug 13  2014 billeder
-rw-r--r--    1 2320585    2322003            53 Aug 11  2014 google87d5eac8
-rw-r--r--    1 2320585    2322003           418 Apr 30  2014 index.php
-rw-r--r--    1 2320585    2322003         19930 Apr 30  2014 license.txt
```

Ovenstående billede viser et udsnit af FTP-serverens filer i det givne Directory.

Der vælges at downloade fil: license.txt

```
**************************************
Indtast 1 for list-kommando
Indtast 2 for retrieve-kommando
Indtast 3 for sensor-menu
Indtast 4 for at afslutte programmet
**************************************

2
**************************************
Hvor skal filen placeres paa harddisken?
Eksempel: C:/Users/JACOB/Desktop/Test/
NB: slut af med /
**************************************

/Users/JacobWorckJepsen/Desktop/test/
**************************************
Skriv navnet paa filen du oensker at hente:
**************************************

license.txt
**************************************
*** Filen hentes ***
**************************************

**************************************
Filen blev hentet.
**************************************
```

Når RETR-kommando vælges, bedes brugeren indtaste hvor filen skal placeres. Her vælges der /User/JacobWorckJepsen/Desktop/test/ - som er et gyldigt Directory. Dernæst skal brugeren skrive hvilken fil der ønskes downloadet. Til slut prompt'es bruger om filoverførslen lykkedes.

***Negativ test** - forkert filnavn:*

Programmet startes op og filen vælges placeret på sti: /Users/JacobWorckJepsen/Desktop/test/, som er et gyldigt Directory. Dernæst downloades fil: blablabla.txt, som ikke eksisterer.

```
****************************************
Hvor skal filen placeres paa harddisken?
Eksempel: C:/Users/JACOB/Desktop/Test/
NB: slut af med /
****************************************

/Users/JacobWorckJepsen/Desktop/test/
****************************************
Skriv navnet paa filen du oensker at hente:
****************************************

blablabla.txt
****************************************
*** Filen hentes ***
****************************************

****************************************
Filen fandtes ikke på serveren.
****************************************
```

Ovenstående billede viser fremgangsmåden ved at hente en fil. Først skrives hvor henne på computeren filen ønskes downlaodet, dernæst filnavnet. Det indtastede filnavn eksisterer ikke og brugeren bliver prompt'et med: "Filen fandtes ikke på serveren".

Vægtprogram

Analyse

Kravspecifikation

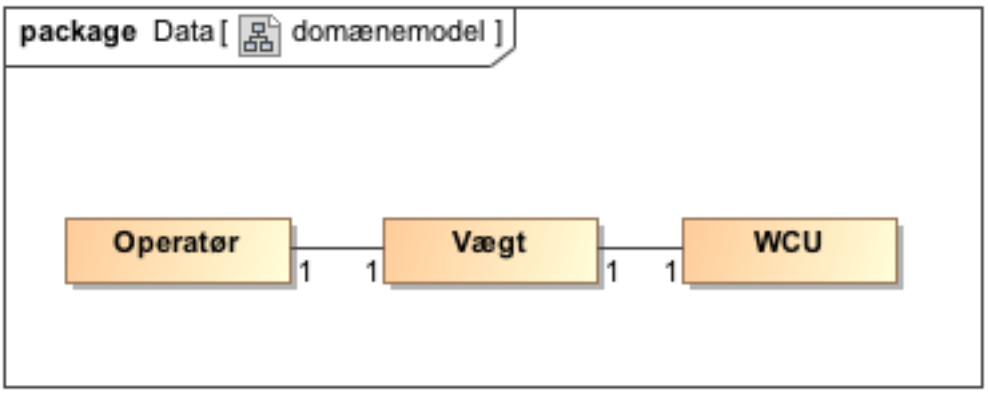
Funktionelle krav

- Systemet skal kunne styre vægten via TCP socket
- Systemet skal kunne håndtere sekvensen i målingen
- Informative fejlmeddelelser udskrives ved betjeningsfejl
- Afvejningsproceduren skal kunne afbrydes når som helst
- Registrere visse ting:
- Store.txt skal indeholde: varenummer, varenavn, varemængde (i kg), det er kun mængde på lager der ændres.
- Log.txt skal indeholde: dato, tid, operatørnummer, varenummer, afvejning (i kg), tilbage på lager (i kg) – historik over betjeninger. Eksisterende data ændres aldrig, der tilføjes udelukkende.
- Operatoer.txt skal indeholde: operatørnumre og navne.
- Fornuftig dialog med operatør via afvejningsproceduren.

Non-funktionelle krav

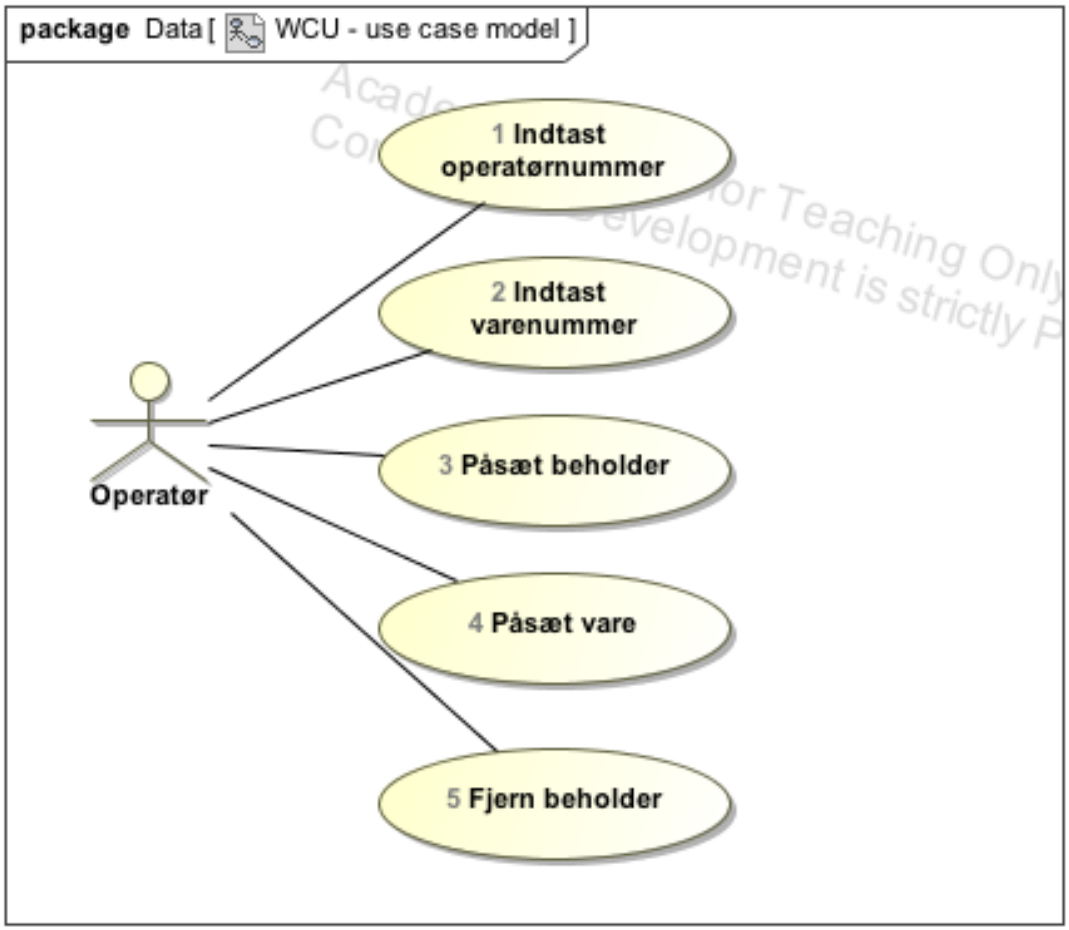
- Fejl skal catches så programmet ikke termineres.
- Instruktioner til operatør skal sendes via RM20
- Programmet arbejder på lokal database (txt filer)

Domæne Model



I ovenstående figur ses domænemodellen for WCU'en. Som domænemodellen viser hvorledes WCU'en kommunikerer med operatøren igennem vægten.

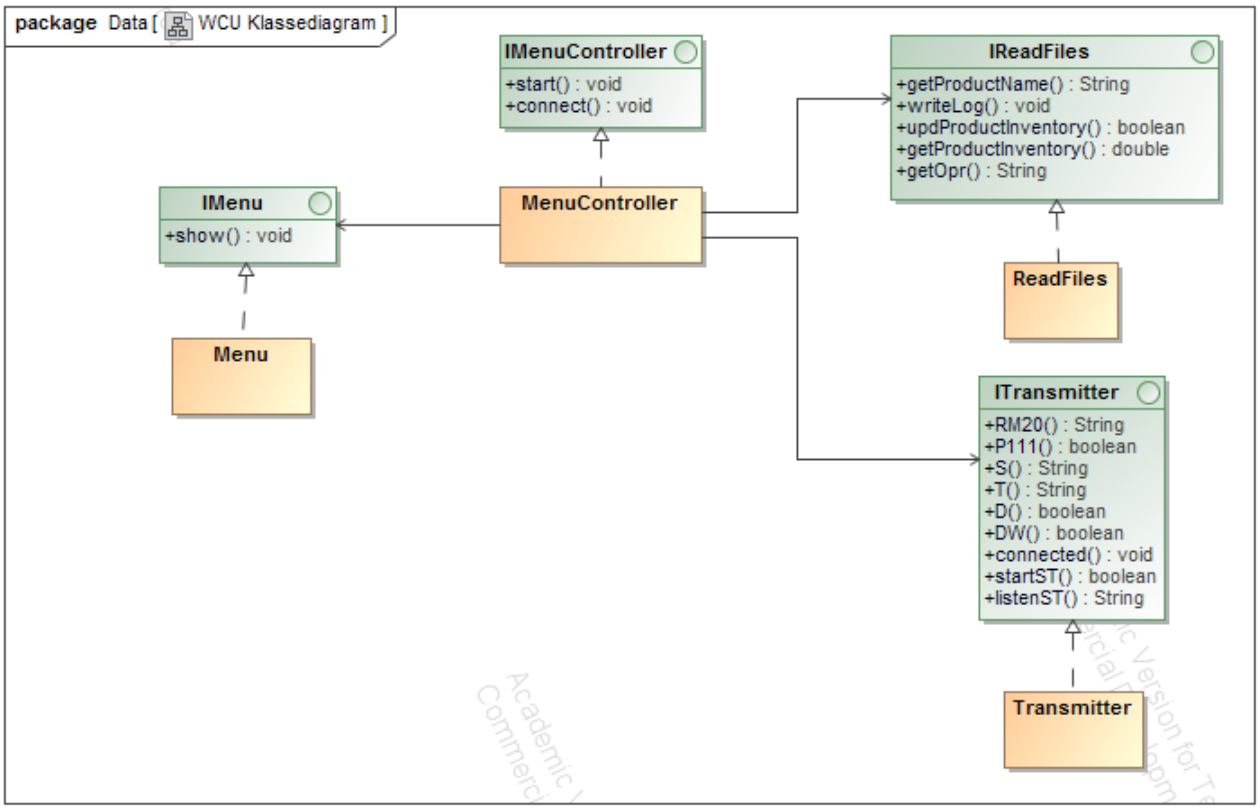
Use Case Model



Ovenstående figur viser de forskellige use-cases der er identificeret i forbindelse med udarbejdelsen af WCU. De er opdelt således at de passer med de stadier som operatøren gennemgår undervejs i en afvejningsproces – og samme stadier som udgør den enum, som indgår i klassen MenuController. De mere detaljerede use case beskrivelser ses i bilag.

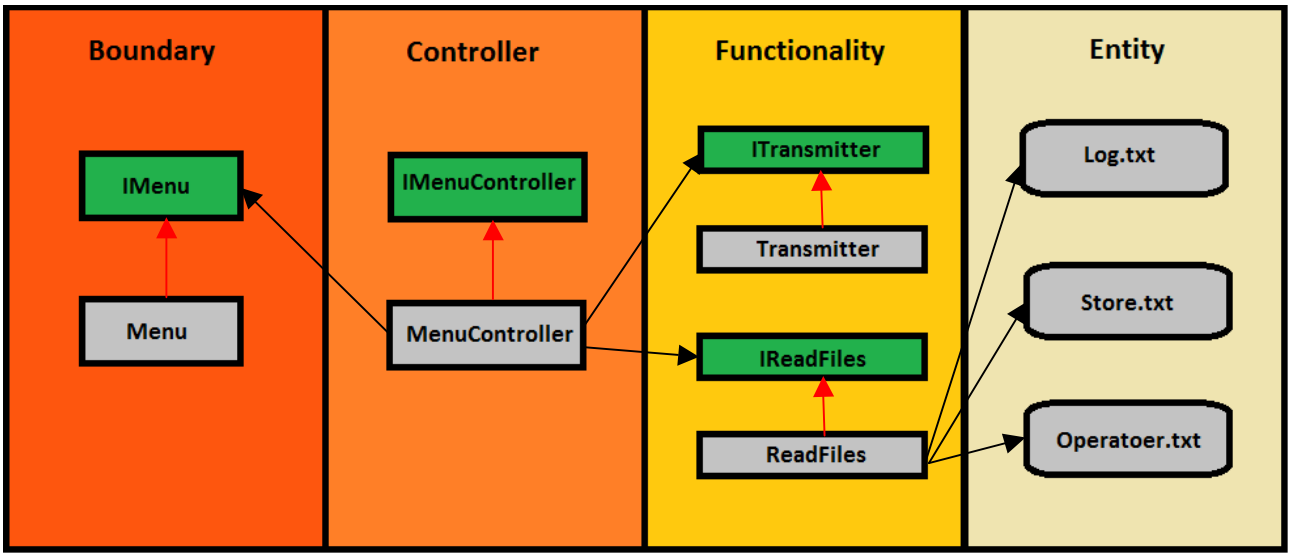
Design

Klassediagram



Ovenstående figur illustrerer design klassediagrammet for den udviklede WCU. For overskuelighedens skyld er det er valgt kun at vise metoderne i de interfaces der benyttes. Ydermere er de parametre metoderne skal modtage udeladt i oversigten. Selve implementationen af de enkelte interfaces er mindre betydende, så længe de opfylder den kontrakt der er beskrevet i javadoc. Implementationen af et interface har dermed som minimum de metoder der er defineret i interfacet, og kan derudover have et antal private metoder der hjælper med den specifikke implementation.

Det er valgt at holde boundary'en i programmet meget simpel, så der kan vises løbende information om afvejningsproceduren. Dette kunne eventuelt udelades i det endelige program (CDIO final), hvor der så til gengæld kunne tilføjes en metode som returnerer hvilken state afvejningsproceduren er i.



Ovenfor ses opbygning af WCU i relation til 3-lagsmodellen inklusiv et controllerlag. Det er værd at bemærke at datalaget Entity i programmet består af .txt-filer, og data ikke er gemt i separate java-klasser. Ligeledes ses det at der i overensstemmelse med design principperne er programmet op mod interfaces for alle lag. Dette holder koblingen på et meget lavt niveau.

Fil-håndtering

Systemet benytter 3 forskellige kommaseparerede filer:

Log.txt

- En løbende log over historikken – hvem har gjort hvad. Filen indeholder *Data + tid, operatørnummer, varenummer, afvejning i kg, restmængde på lager*. Der overskrives ikke data i loggen, der tilføjes udelukkende data.

Store.txt

- Holder status på lageret. Indeholder *varenummer, varenavn, varemængde i lager*. Varemængden opdateres hver gang der bruges noget materiale fra lageret.

Operatoer.txt

- Indeholder information om de oprettede operatører – *operatørnummer, navn, cpr-nummer*. Filen opdateres ikke, men metoden der læser filen er robust nok til at kunne håndtere at yderligere operatører tilføjes.

```
120     private String[] readFile(File fil) throws FileNotFoundException{
121         List<String> data = new ArrayList<String>();
122         String linje = null;
123         try (BufferedReader br = new BufferedReader(new FileReader(fil));){
124             while ((linje = br.readLine()) != null){
125                 data.add(linje);
126             }
127         } catch (IOException e) {
128             throw new FileNotFoundException();
129         }
130         return data.toArray(new String[data.size()]);
131     }
```

Alle filer læses via den ovenstående metode *readFile(File fil)* i ReadFiles.java. Metoden læser en fil en linje af gangen (linje 124), og sætter den læste linje ind i en ArrayList af Strings (linje 125). Dette gøres så længe der er linjer i tekstfilen.

Såfremt der opstår en fejl i læsningen af filen kastes en FileNotFoundException som fanges i Main.java. Her udskrives blot en simpel besked om at fillæsningen er fejlet, og at brugeren bør kontakte udvikleren.

Til slut konverteres ArrayListen til et array af Strings og dette array returneres (linje 130).

```
104     @Override
105     public double getProductInventory(int productNumber) throws FileNotFoundException{
106         double out = -1;
107         String[] data = readFile(store);
108         int p = 0;
109         while (p < data.length){
110             String[] line = data[p].split(",");
111             if (productNumber == Integer.parseInt(line[0])){
112                 out = Double.parseDouble(line[2]);
113             }
114             p++;
115         }
116         return out;
117     }
```

Ovenstående figur viser hvordan fillæsningen benyttes når lagermængden for et produkt skal findes. Metoden String.split(",") benyttes til at opsplitte hver String i arrayet data i nye Strings for hvert komma. Herefter kan vi benytte at vi kender formateringen i de kommaseparerede filer og i dette tilfælde ved at produktnummeret er det første tal/String i hver linje, samt at lagerstatus er den tredje (dvs. plads 0 og 2 i arrayet).

```
131     private void writeFile(File fil, String[] data) throws FileNotFoundException{
132         try (BufferedWriter write = new BufferedWriter(new FileWriter(fil));){
133             for (String s : data){
134                 write.write(s);
135                 write.newLine();
136             }
137         } catch (IOException e) {
138             throw new FileNotFoundException();
139         }
140     }
```

Det er valgt at lagerstatus for hver enkelt vare løbende opdateres når en vare bliver brugt i processen. Dette opdateres i store.txt, ved at læse hele filen, ændre varemængden for det

pågældende produkt, og til slut overskrive store.txt med det nye data. Det er dermed en ret ineffektiv metode til at ændre et enkelt tal i en fil. Til dette projekt er det dog valgt at beholde denne løsning, idet det endelige projekt vil indeholde en database i stedet for store.txt, hvor det nemmere lader sig gøre at ændre en enkelt række i en tabel i stedet for at skrive hele databasen for hver gang et produkt bruges.

Procedure-håndtering

Til at styre den identificerede procedure er der blevet defineret en variabel (state) af den specielle datatype enum (State). Denne variabel kan udelukkende antage prædefinerede værdier og hver værdi er valgt til at svare til et punkt i proceduren. Dog er værdien STOP ikke direkte implementeret, den bruges udelukkende til at holde menuen kørende. Operatøren har ikke adgang til STOP da procedureprogrammet ikke skal kunne stoppes fra vægten. Svarende til at en klient ikke kan stoppe et serverprogram. Værdierne, med tilhørende procedurehåndteringer, er:

- START *(Start procedure, OperatørID, bekræft oprID)*
- GET_PROD_NR *(Produktnummer, bekræft produktnavn)*
- SET_CONTAINER *(Påsæt beholder, tarering)*
- ADD_PRODUCT *(Afvej vare, aflæs vægt)*
- REMOVE_CONTAINER *(Fjern beholder, log af vareforbrug, log af afvejningsinstans)*
- RESTART *(Bekræft ny afvejning, genstart procedure)*
- STOP *(bruges ikke til selve proceduren)*

Enum er speciel da de prædefinerede værdier kan indeholde variabler og metoder, enumværdier agerer som objekter. I enumtypen State er der defineret to abstrakte metoder desc() og changeState(…), som alle de deklarerede værdier heri skal implementere i stil med implementering af et interface i en klasse. Desc() bruges til at udskrive navnet på den pågældende værdi som debug i konsollen. ChangeState(…) indeholder logikken bag hvert skridt i proceduren repræsenteret af de individuelle værdier. Således kan changeState(…) kaldes på variablen state og alt efter hvilken værdi state har bliver den korresponderende changeState(…) metode kaldt. På denne måde kan der nemt skiftes frit mellem de forskellige skridt i proceduren ved blot at have en enkelt linje kode i et loop, frem for en mindre elegant løsning med mere simple datatyper.

```
177        ADD_PRODUCT {
179            String desc() {..
182            @Override
183            State changeState(IMenu menu, IReadFiles fileAccess, ITransmitter trans, IMenuController Imc) {
184                String input = null, answer = "OK";
185                try{
186                    input = trans.RM20("Afvej vare og kvittér:","OK","?");
187                    if(input.toLowerCase().equals("q")){
188                        return STOP;
189                    } else if (input.equals(answer)) {
190                        trans.P111("Efter vejning, kvittér med dør-knap");
191                        trans.startST(true);
192                        Imc.setAfvejning(Double.parseDouble(trans.listenST()));
193                        trans.startST(false);
194                        trans.P111("");
195                        return REMOVE_CONTAINER;
196                    } else {
197                        return ADD_PRODUCT;
198                    }
199
200                } catch (NumberFormatException | IOException e) {
201                    menu.show("Forkert type input. Prøv igen");
202                    return ADD_PRODUCT;
203                }
204            }
205        },
```

Metodeimplementering i værdien ADD_PRODUCT.

```
49      @Override
50      public void start(){
51          menu.show("Overvågning af vægtbetjening");
52          do{
53              menu.show("");
54              menu.show(state.desc());
55              this.state = this.state.changeState(menu,fileAccess,trans,this);
56          }
57          while(!state.equals(State.STOP));
58      }
```

Loop hvori skift af enum kører indtil variablen antager værdien STOP.

Test

For at teste WCU programmet er der dokumenteret en positiv bruger test. Herunder følger testen delt op på de individuelle skridt i proceduren. Ved hvert skridt er det forventelige flow dokumenteret, efterfulgt af en gennemgang af de fejlbehæftede alternativer, der er taget højde for. Testen er udført på en simulator af den rigtige vægt, så der vil eksistere mindre forskelle. Efterfølgende er programmet blevet brugertestet på selve vægten og resultatet er modsvarer det dokumenterede.

START

Korrekt flow:

```
State: START
Indtast operatørnummer:
13
Bruger valgt: Jon Tvermose. Er dette
korrekt?
Jon Tvermose bekræftet.
```

```
****************** Display ********************
Bekræft bruger:
Jon Tvermose  ?                          0,0000

***********************************************

***************** Debug info ******************
Tilknyttet port: 8000
Brutto: 0.0 kg
Tara: 0.0 kg
Seneste modtagne kommando: RM20 8 "Bekræft bruger:" "Jon Tvermose" " ?"
Seneste afsendte svar: RM20 B
***********************************************

Indtast (T/B/Q for taratryk / brutto ændring / quit)
Tast her:
```

Alternativer (operatør ID):

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"
Ukendt positivt heltal (2) -> "Bruger findes ikke. Prøv igen."
Negativt heltal (-2) -> "Forkert input type. Prøv igen."
Karakter eller streng (test) -> "Forkert input type. Prøv igen."
Tom ( ) -> "Forkert input type. Prøv igen."

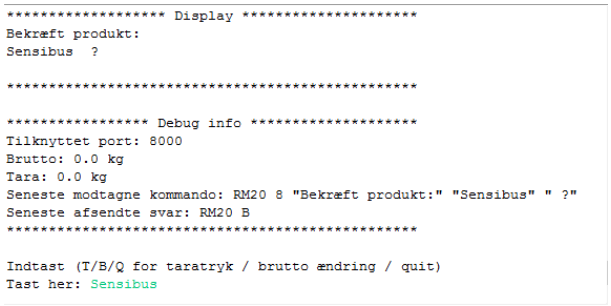
Alternativer (bekræft bruger):

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"
Bruger ikke bekræftet (nej) -> "Forkert bruger. Prov igen."
Ukendt positivt heltal (2) -> "Bruger findes ikke. Prov igen."
Negativt heltal (-2) -> "Forkert input type. Prøv igen."
Karakter eller streng (test) -> "Forkert input type. Prøv igen."
Tom ( ) -> "Forkert input type. Prøv igen."

GET_PROD_NR:

Korrekt flow:

```
State: GET_PROD_NR
Indtast varenummer:
7
Sensibus
Produkt valgt: Sensibus. Er dette
korrekt?
Produkt bekræftet.
```



Alternativer (varenummer):

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"
Ukendt positivt heltal (2) -> "Produkt findes ikke. Prov igen."
Negativt heltal (-2) -> "Forkert input type. Prøv igen."
Karakter eller streng (test) -> "Forkert input type. Prøv igen."
Tom ( ) -> "Forkert input type. Prøv igen."

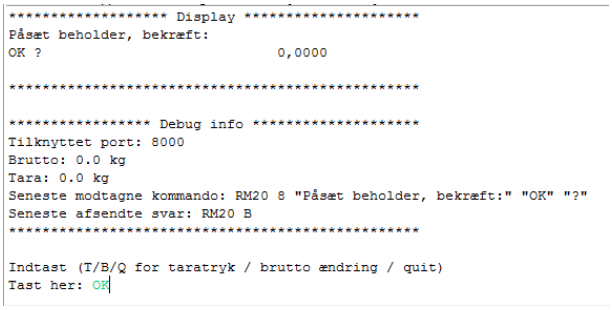
Alternativer (bekræft produkt):

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"
Produkt ikke bekræftet (nej) -> "Forkert produkt. Prov igen."
Ukendt positivt heltal (2) -> "Forkert produkt. Prov igen."
Negativt heltal (-2) -> " Forkert produkt. Prøv igen."
Karakter eller streng (test) -> "Forkert produkt. Prøv igen."
Tom ( ) -> " Forkert produkt. Prøv igen."

SET_CONTAINER:

Korrekt flow:

```
State: SET_CONTAINER
Påsæt beholder og bekræft.
OK
Beholder påsat
Vægt tareret: 0.0
```



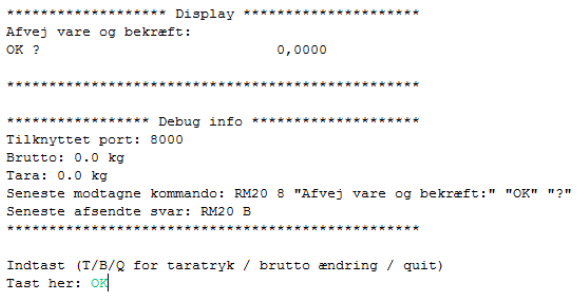
Alternativer:

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"
Ukendt positivt heltal (2) -> "Beholder ej pasat. Prov igen."
Negativt heltal (-2) -> "Beholder ej pasat. Prøv igen."
Karakter eller streng (test) -> "Beholder ej pasat. Prøv igen."
Tom ( ) -> "Beholder ej pasat. Prøv igen."

ADD_PRODUCT:

Korrekt flow:

```
State: ADD_PRODUCT
Afvej vare og bekræft.
OK
Afvej og kvittér med dør-knap
1.2519 afvejet.
```



Alternativer:

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"

Ukendt positivt heltal (2) -> "Vare ej afvejet. Prov igen."

Negativt heltal (-2) -> "Vare ej afvejet. Prøv igen."

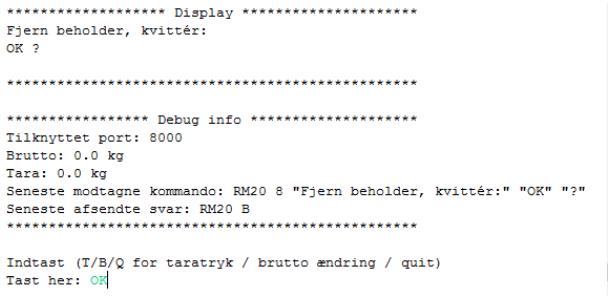
Karakter eller streng (test) -> "Vare ej afvejet. Prøv igen."

Tom ( ) -> "Vare ej afvejet. Prøv igen."

REMOVE_CONTAINER:

Korrekt flow:

```
State: REMOVE_CONTAINER
Fjern beholder og bekræft.
OK
Beholdning opdateret:
Vare ID: 7, Afvejning: 1.2519
Log skrevet:
Operatør ID: 13, Vare ID: 7, Tara vægt:
0.0, Afvejning: 1.2519
```



Alternativer:

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"

Ukendt positivt heltal (2) -> "Beholder ej fjernet. Prov igen."

Negativt heltal (-2) -> "Beholder ej fjernet. Prøv igen."

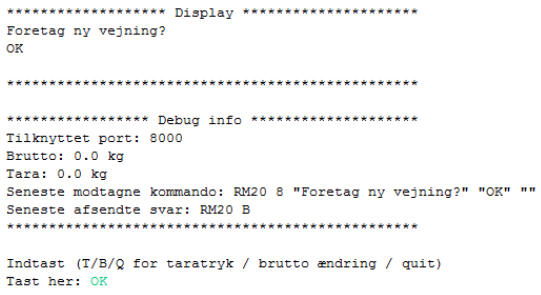
Karakter eller streng (test) -> "Beholder ej fjernet. Prøv igen."

Tom ( ) -> "Beholder ej fjernet. Prøv igen."

RESTART:

Korrekt flow:

```
State: RESTART
Foretag ny vejning?
OK
Proceduren genstartes.
```



Alternativer:

Bruger afbrydelse (q) -> "Proceduren afbrudt af brugeren"

Ukendt positivt heltal (2) -> "Proceduren afbrudt af brugeren"

Negativt heltal (-2) -> "Proceduren afbrudt af brugeren"

Karakter eller streng (test) -> "Proceduren afbrudt af brugeren"

Tom ( ) -> "Proceduren afbrudt af brugeren"

Projektafslutning

Der er i projektet blevet oprettet en FTP-klient der kan oprette forbindelse, samt sende og modtage kommandoer. Ydermere kan der downloades filer fra serveren, som kan placeres i de rigtige mapper på computeren. FTP-klienten er simplificeret og er bygget fra grunden af, med rigtig syntaks og formatering. Dette betyder at de fleste FTP-servere kan modtage og forstå kommandoer der kom fra klienten.

Ydermere er der lavet en forbindelsesfunktion til sensorerne der er på zybo-boardet, samt en menu til at vælge sensorerne. Der er i den forbindelse udviklet en protokol baseret på en række antagelser. Det har ikke været muligt at teste protokollen op imod et Zyboboard.

Der er udviklet en WCU som kan håndtere en afvejningsprocedure samt adskillige fejlsituationer som beskrevet i testafsnittet. WCUen er designet til at fungere med en af de fysiske vægte. Hvis WCUen tilsluttes en vægtsimulator er det derfor yderst kritisk at syntaksen på svar på kommandoer fra vægtsimulatoren er identiske med dem fra den fysiske vægt.

Mht. filhåndteringen er der valgt at løse de frivillige opgaver at opdatere en varemængde på lageret, samt at indlæse et antal operatører fra en fil.

Bilag

Kildehenvisninger

- http://www.java2s.com/Code/Java/Network-Protocol/ImplementsaJavaFTPclientfromsocketandRFC.htm - Brugt til implementering af FTP-klient.

FTP

Use case navn: Opretter forbindelse til server

ID: 1

Kort beskrivelse:
Opretter forbindelse til FTP server, ved hjælp af en socket forbindelse.

Primære aktører
Forbrugeren

Sekundære aktører
FTP Server

Preconditions

- At der oprettes en socket forbindelse på port 21.
- Username og Password til FTP server er korrekt.

Main flow

1. klienten opretter forbindelse til FTP serverens IP adresse på port 21.
2. Serveren sender kommandoen 200 - ok der er forbindelse
3. klienten sender kommandoen USER username til serveren
4. Serveren svare med kommandoen 230- USER username ok - password kræves
5. klienten sender kommandoen PASS password til serveren
6. Serveren sender kommandoen 230 ok du er nu logget ind

Postconditions

-Forbindelsen er oprettet mellem klienten og serveren på port 21.

Alternativ flows

1. klienten kan ikke oprette forbindelse til FTP serveren, får fejlmeddelelsen ” `"FTP klienten modtog ukendt respons ved oprettelse af forbindelse til server: "`
2. Forbrugeren indtaster ved en fejl forkert username eller password og får fejlmeddelelsen `"FTP klienten modtog forkert respons fra server efter brugernavn blev indtastet: "`

Use case navn: Liste kommando

ID: 2

Kort beskrivelse:
Sender en kommando til FTP serveren, som spørger hvilke filer der ligger på serveren.

Primære aktører
Forbrugeren

Sekundære aktører
FTP Server

Preconditions

- At der er forbindelse til FTP serveren

Main flow

7. klienten sender en kommando til serveren om hvilke filer der ligger på FTP serveren.
8. Serveren sender en liste tilbage med hvilke filer der ligger på serveren.
9. Klienten printer listen ud på et display.

Postconditions

-Klienten viser på et display, en liste over mulige filer der kan hentes.

Alternativ flows

Use case navn: Retrive kommando

ID: 3

Kort beskrivelse:
Vælger hvilken fil man vil download fra FTP serveren

Primære aktører
Forbrugeren

Sekundære aktører
FTP Server

Preconditions

- At der er forbindelse til FTP serveren
- At man er logget ind med en bruger, som bliver godkendt af serveren

Main flow

10. klienten sender en kommando til serveren om hvilke fil der vil downloades(kommandoen er: RETR navn_på_fil.filformat.
11. klienten og Serveren opretter en ny socket forbindelse, hvor serveren sender filen til klienten.

Postconditions

-At den valgte fil, er blevet hentet og ligger på ens harddisk.

Alternativ flows

1. Forbrugeren indtaster et navn på en fil der ikke eksisterer på serveren, og får fejlen `"Filen eksisterer ikke"`

Use case navn: Øge sit samplingsinterval

ID: 4

Kort beskrivelse:
Får sensor 222 på zybo til at øge sit samplingsinterval

Primære aktører
Forbrugeren

Sekundære aktører
Sensor 222

Preconditions

- At der er forbindelse til zybo

Main flow

12. Klienten sender en kommando til zybo.
13. Zybo sender kommando til sensor 222
14. Sensor 222 stopper med at måle

Postconditions

-Sensor 222 stopper med at måle

Alternativ flows

Use case navn: Afslutte programmet

ID: 5

Kort beskrivelse:
Lukker programmet ned.

Primære aktører
Forbrugeren

Sekundære aktører

-

Preconditions:

- At programmet kører

Main flow

15. Forbrugeren indtaster tallet 4.
16. Programmet afsluttes

Postconditions

-Programmet bliver lukket ned

Alternativ flows

Use case navn: Øge sit samplingsinterval

ID: 6

Kort beskrivelse:
Får sensor 1 på zybo til at øge sit samplingsinterval

Primære aktører
Forbrugeren

Sekundære aktører
Sensor 1

Preconditions

- At der er forbindelse til zybo

Main flow

17. Klienten sender en kommando til zybo.
18. Zybo sender kommando til sensor1
19. Sensor1 øger sit samplingsinterval

Postconditions

-Sensor 1 øger sit samplingsinterval

Alternativ flows

Use case navn: Øge sit samplingsinterval

ID: 7

Kort beskrivelse:
Får sensor 4 på zybo til at mindske sit samplingsinterval

Primære aktører
Forbrugeren

Sekundære aktører
Sensor 4

Preconditions:

- At der er forbindelse til zybo

Main flow

20. Klienten sender en kommando til zybo.
21. Zybo sender kommando til sensor 4
22. Sensor 4 mindsker sit samplingsinterval

Postconditions

-Sensor 4 mindsker sit samplingsinterval

Alternativ flows

Use case navn: Øge sit samplingsinterval

ID: 8

Kort beskrivelse:
Får sensor 7 på zybo til at måle

Primære aktører
Forbrugeren

Sekundære aktører
Sensor 7

Preconditions

- At der er forbindelse til zybo

Main flow

23. Klienten sender en kommando til zybo.
24. Zybo sender kommando til sensor 7
25. Sensor 7 begynder at måle

Postconditions
- Sensor 7 begynder at måle

Alternativ flows

WCU

Use case navn: Indtast operatørnummer

ID: 1

Kort beskrivelse:
Operatør indtaster operatørnummer i vægt.

Primære aktører:
Operatør

Sekundære aktører:

-

Preconditions:

- Der er forbindelse mellem vægt og WCU

Main flow:

1. Displayet i vægten viser: ”Indtast bruger ID (10-16):”
2. Operatør indtaster bruger ID 11
3. Displayet i vægten viser: ”Korrekt bruger: Christian Mikkelsen? ”
4. Operatør bekræfter på vægt

Postconditions:

- WCU går videre til Use Case 2 – udfyld varenummer

Alternative flows:

1. Displayet i vægten viser: ”Indtast bruger ID (10-16):”
2. Operatør indtaster bruger ID 11v
3. Displayet viser: ”Forkert input type. Prøv igen”
4. Displayet i vægten viser: ”Indtast bruger ID (10-16):”

Use case navn: Indtast varenummer

ID: 2

Kort beskrivelse:
Operatør indtaster varenummer i vægt

Primære aktører:
Operatør

Sekundære aktører:

-

Preconditions:

- Use case 1 – Indtast operatørnummer

Main flow:

5. Displayet i vægten viser: "Tast produkt ID (1-9):"
6. Operatør indtaster varenummer "5"
7. Displayet i vægten viser: "Bekræft produkt: Voluptaria?"
8. Operatør bekræfter på vægt

Postconditions:

- WCU går videre til Use Case 3 – Påsæt skål

Alternative flows:

1. Displayet i vægten viser: "Tast produkt ID (1-9):"
2. Operatør indtaster "z"
3. Displayet viser: "Forkert input type. Prøv igen"
4. Displayet i vægten viser: "Tast produkt ID (1-9):"

Use case navn: Påsæt beholder

ID: 3

Kort beskrivelse:
Operatør bedes om at påsætte beholder på vægt

Primære aktører:
Operatør

Sekundære aktører:

Preconditions:

- Use case 1 – Indtast operatørnummer
- Use case 2 – Indtast varenummer

Main flow:

1. Display viser: ” "Påsæt beholder, kvittér: OK?”
2. Operatør påsætter beholder
3. Operatør bekræfter
4. WCU tarerer vægten.

Postconditions:

- WCU går videre til Use Case 4 – Tilføj vare

Alternative flows:

-

Use case navn: Tilføj vare

ID: 4

Kort beskrivelse:
Operatør bedes om at tilføje vare

Primære aktører:
Operatør

Sekundære aktører:

Preconditions:

- Use case 1 – Indtast operatørnummer
- Use case 2 – Indtast varenummer
- Use case 3 – Påsæt beholder

Main flow:

1. Display viser: ” Afvej vare og kvittér:","OK","?"
2. Operatør bekræfter med ok
3. Display viser: "Efter vejning, kvittér med dør-knap"
4. Operatør afvejer vare
5. Operatør bekræfter med dør-knap.

Postconditions:

- WCU går videre til Use Case 5 – Fjern beholder

Alternative flows:

-

Use case navn: Fjern beholder

ID: 5

Kort beskrivelse:
Operatør bedes om at fjerne beholder fra vægt

Primære aktører:
Operatør

Sekundære aktører:

Preconditions:

- Use case 1 – Indtast operatørnummer
- Use case 2 – Indtast varenummer
- Use case 3 – Påsæt beholder
- Use case 4 – Tilføj vare

Main flow:

1. Display viser: "Fjern beholder, kvittér:","OK"
2. Operatør fjerner beholder
3. Operatør bekræfter med OK

Postconditions:

- WCU opdaterer produktliste med den fjernede mængde produkt

- WCU registrerer afvejningen i loggen.

- Vægt er nu klar til ny afvejning

Alternative flows:

-